Symbolic Analysis of an Electric Vehicle Charging Protocol

Li Li*, Jun Pang†, Yang Liu‡, Jun Sun§, Jin Song Dong*
*School of Computing, National University of Singapore, Singapore
†FSTC and SnT, University of Luxembourg, Luxembourg
‡School of Computer Engineering, Nanyang Technological University, Singapore
§Information System Technology and Design, Singapore University of Technology and Design, Singapore

***Abstract*—In this paper, we describe our analysis of a recently proposed electric vehicle charing protocol. The protocol builds on complex cryptographic primitives such as commitment, zero-knowledge proofs, BBS+ signature and etc. Moreover, interesting properties such as secrecy, authentication, anonymity, and location privacy are claimed on this protocol. It thus presents a challenge for formal verification, as no single existing tool for security protocol analysis support for all the required features. In our analysis, we employ and combine the strength of two state-of-the-art symbolic verifiers, Tamarin and ProVerif, to check all important properties of the protocol.**

I. Introduction

Electric vehicles are promising and futuristic automobiles which use electric batteries for clean energy. They dominate conventional vehicles from several aspects such as air pollution reductions, less power emissions and lower oil dependencies. In addition, Vehicle-to-Grid (V2G) is proposed to balance electricity load for electric system. In V2G, it is possible and highly recommended to do charging when the demand is low, especially after midnight, and to send the electricity back (recharging) to the system during the peak time. Despite these advantages, one of the concerns is the potential privacy leakage along with the charging route. Since energy storage devices nowadays cannot meet the requirement for long-term driving, electric vehicles need to visit charging stations frequently for energy supplying. As a consequence, the location privacy disclosed along with the charging and recharging behaviors has drawn particular attentions.

Due to this specific application requirement, a privacy-preserving electric vehicle charging protocol (ECP) has been recently designed by Liu et al. [1]. In this protocol, various complex cryptographic primitives are used and many security properties are claimed. Firstly, the user could make *commitment* to some data and expose the key later to open the commitment. It requires that the secrecy properties of the keys are related to the order of events happened in the protocol. Specifically, the commitment scheme requires that the private opening keys for the commitments should be unknown to the supplier until the user exposes them explicitly in the protocol. Secondly, this protocol supports the *two-way transmission*, which means the users are allowed to charge their vehicle at the stations, as well as recharge the electricity back to the stations with their balance refunded. This is achieved using multiple generators in the commitment scheme which is *homomorphic*, such that operations could be made on commitment to change the balance without knowing the explicit value. In addition, the supplier is potentially *dishonest* in this protocol. *Injective agreement* between the user and the supplier should always be guaranteed, which ensures that the supplier could only charge the users just as he should. Fourthly, the protocol is *stateful* in which manipulations over global mutable state are required. In this protocol, each user gets an account state after the registration. He needs to use the information stored in the state to communicate with the supplier in the later sessions and update them after each successful transaction.[1] Lastly, privacy properties such as anonymity and location privacy should be preserved by the protocol against the supplier. Anonymity makes sure that the supplier could not get any partial information about the users when they charge and recharge at the stations. Meanwhile, location privacy ensures that the supplier could not identify the station where users perform charing or recharging. In this protocol, privacy properties are achieved by using the zero-knowledge proofs and BBS+ signature [2] with commitment scheme.

Even though most of security protocol are designed carefully and manual proofs are given along with their publication, the protocols as well as their proofs are still proven to be error-prone. This can be well illustrated by the famous Needham-Schroeder public-key protocol [3], in which a security flaw was found by Gavin Lowe 17 years after its publication [4]. Therefore, automatic verification is very helpful for ensuring the correctness or finding attack on security protocols. In this paper, we thoroughly perform a formal analysis for the electric vehicle charging protocol [1]. As many symbolic tools, such as Scyther [5], Tamarin [6], ProVerif [7], StateVerif [8], and Athena [9] have been developed for automatic analysis of security protocols using different approaches, selecting the right techniques and tools to verify a complex protocol such as [1] is non-trivial. Due to the number of security and privacy properties claimed by the protocol, no single protocol verifier could give a complete verification of the protocol at present. Thus, we combine the verification capacities from Tamarin and ProVerif, to give a thorough verification of the protocol: Tamarin [6] can handle stateful protocols naturally and allows

[1] As a result, some security protocol checkers could not terminate or even specify this protocol because of the infinite execution trace involved.

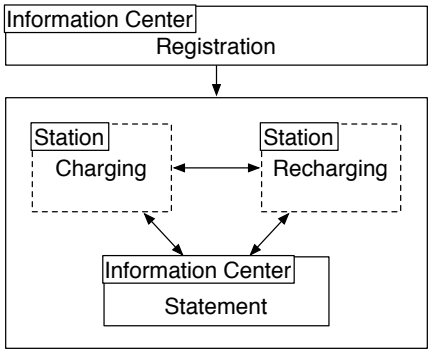

Fig. 1. Protocol Overview

us to check event order related secrecy and authentication properties, while ProVerif [7] can check observational equivalence [10] so that we use it for checking privacy properties.

Our contributions. First, we present a study of a few state-of-the-art symbolic tools for security protocol analysis and discuss their strength and weakness. We then propose to combine the verification capacities of Tamarin and ProVerif to analyze the electric vehicle charging protocol, in which secrecy, authentication and privacy properties all play important roles. To the best of our knowledge, this is the first work to use Tamarin and ProVerif together for analyzing a single protocol. We formally define the properties claimed by the protocol and give their verification results in Tamarin and ProVerif.

Structure of the paper. In Section II, we present the structural overview for this protocol, the cryptographic primitives used in the protocol and the properties required by this protocol. We then describe and compare four state-of-the-art security protocol verifiers in details and address the reasons why we choose ProVerif and Tamarin as the verification tools for this protocol in Section III. In Section IV, we will illustrate the modeling in Tamarin along with the verification for secrecy and authentication properties. In Section V, we will show how we specify protocol in ProVerif as well as the anonymity and privacy properties checked. Finally, we draw conclusions and discuss about the future work in Section VI

II. The Protocol

In this paper, we perform formal verification for the properties claimed by an electronic vehicle charging protocol [1]. This protocol is designed to protect the users' privacy, e.g., anonymity and unlinkability, against the supplier even when they frequently charge and recharge at the stations. It consists of four sub-routines such as registration, charging, recharging and statement, whose overall structure is shown in Figure 1. [2]

Registration. Firstly, the users need to register at the information center. They show their identities to the supplier and pay a default deposit to open their accounts. The supplier then give each user a token in return with his balance embedded inside.

Charging and Recharging. During the charging and recharging phases, the users provide their token to the station in an anonymous way. After checking the token, the station will update the token and send it back to the user. Since the station can record the information he receives from the users, the charging and recharging transactions should be taken in a partially blind way.

Statement. When the balance in the user's account is running low, the users could go to the information center again to disclose their identities and top-up money directly into their accounts.

A. Cryptographic Primitives

To achieve the desired properties, various cryptographic primitives have been used in the protocol, including commitment, zero knowledge proof, and BBS+ signature combined with bilinear pairing.

Commitment. This protocol used the well known commitment scheme developed by Pedersen [11], in which a secret opening value t is chosen randomly by the committer to compute the commitment c over a tuple of public values $(x_0, x_1, \ldots, x_n)$ as $C(x_0, x_1, \ldots, x_n, t) = g_0^{x_0} g_1^{x_1} \ldots g_n^{x_n} g_{n+1}^{t}$. t is unknown to the public at first. When the committer wants to prove that c is the commitment for $(x_0, x_1, \ldots, x_n)$ he made, he could reveal the opening value t so that everyone can test if the following equation holds

$$C(x_0, x_1, \ldots, x_n, t) \stackrel{?}{=} c.$$

It has been proven that given a commitment c, no polynomial algorithm exists to find the opening value t with respect to $(x_0, x_1, \ldots, x_n)$ [12], which means only the committer could open the commitment c. Since the commitment scheme is homomorphic, the multiplying of two commitments will give a new commitment with its opening value equals to the sum of the openings from the formers. In this protocol, the commitment is constructed on the basis of four independent generators in a cyclic group $\mathbb{G}$ denoted as $g_1, g_2, g_3, g_4 \in \mathbb{G}$ and we have:

$$C(x_0, x_1, x_2, t) = {g_0}^{x_0} {g_1}^{x_1} {g_2}^{x_2} {g_3}^{t}$$

Zero-Knowledge Proof. As the protocol needs to preserve the privacy properties, the user could not send the datas such as signatures, balances and etc. to the supplier directly. As a result, several zero-knowledge proofs are used in this protocol. Zero knowledge proof protocols generally could be represented by $PK\{(s_0, \ldots, s_n) : f_0 \wedge \ldots \wedge f_m\}$ where PK is the proof given, $(s_0, \ldots, s_n)$ are the private informations and $f_0 \ldots f_m$ are the targets the prover wants to prove. As we treat the zero knowledge proofs as black boxes in the verification, we modeled them as functions with two parts of information. One is the public information pub and another is the private information pvt. The prover generates the proof based on both

[2] The dash rectangles represent the anonymous charging and recharging operations taking place at the stations.

of the public and private information denoted as $prf(pub, pvt)$. To check the proof, the verifier feeds the public information and the proof into the verification function in the form of

$$verif(pub, prf(pub, pvt)) = true,$$

which will return true only when they are correctly matched. Because the prover could give the proof only if he knows the private information, the verifier could then be sure that the prover has the private information he claimed.

BBS+ Signature with Bilinear Pairing. In this protocol, the *BBS+ signature* proposed by Au et al. [2] is employed. It uses an additional generator $g \in \mathbb{G}$. The *BBS+ signature* allows the signer to produce signature in a partially blind way. Specifically, the signer could compute a signature denoted as $A(c, e, r)$ over a commitment c without knowing the values encoded in c by

$$A(c, e, r) = (g * c)^{\frac{1}{e+r}}, \tag{1}$$

where e is a fresh random number, r is the long-term private signing key of the signer, and $w = g^r$ is the respective long-term public verification key. To verify the signature, a bilinear pairing function $\hat{e}$ is employed because of its feature such as $\hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab}$. As a result, everyone could verify the signature by testing

$$\hat{e}(A(c, e, r), w * g^e) \stackrel{?}{=} \hat{e}(g * c, g). \tag{2}$$

B. Protocol Overview

We give detailed descriptions for the four sub-routines of the protocol as follows.

Registration. Before doing payment at the stations, the users need to go to the information center to create their account. At the information center, each user discloses his identity I and pays a fixed deposit D to the supplier. Additionally, he freshly chooses random numbers y', s and sends $c_0 = g_0^{y'} g_3^s$ along with a zero knowledge proof $p = PK_1\{(y', s) : c_0 = g_0^{y'} g_3^s\}$ to the supplier. After receiving I, c_0 and p, the supplier checks the identity and verifies the proof p and computes $A = (c_0 g g_0^{y''} g_1^I g_2^D)^{\frac{1}{e+r}}$ according to (1) where y'' and e are fresh random numbers. Since the opening of $g_0^{y''} g_1^I g_2^D$ is 0, the opening of the commitment encoded in A is the same as of c_0. When the user gets A, y'' and e from the supplier, he could verify the signature using (2) with $c = c_0 g_0^{y''} g_1^I g_2^D$. If the signature is verified successfully, the user stores the tuple $(A, e, y' + y'', I, D, s)$ for later operations.

Charging. Charging operations are conducted at the stations with users' privacy preserved. Initially, the user has the tuple $(\tilde{A}, \tilde{e}, \tilde{y}, I, \tilde{B}, \tilde{s})$ stored in his device. Firstly, the user randomly picks y' and s to compute the commitment $c_0 = g_0^{y'} g_1^I g_2^{\tilde{B}} g_3^s$ and sends it as well as $\tilde{s}$ to the supplier together with a zero knowledge proof $p = PK2\{(\tilde{A}, \tilde{e}, \tilde{y}, I, \tilde{B}, y', s) : c_0 = g_0^{y'} g_1^I g_2^{\tilde{B}} g_3^s \wedge \hat{s}(\tilde{A}, w g^{\tilde{e}}) = \hat{e}(g g_0^{\tilde{y}} g_1^I g_2^{\tilde{B}} g_3^{\tilde{s}}, g) \wedge D \geq \tilde{B} - v \geq 0\}$. After receiving c_0, $\tilde{s}$ and p, the supplier checks that $\tilde{s}$ has never been used and the proof is correct. If the checking is passed, the supplier picks random numbers y'' and e and computes the signature $A = (c_0 g g_0^{y''} g_2^{-v})^{\frac{1}{e+r}}$ where v is the amount of electricity charged. As can be seen, the new commitment encoded in A equals to $g_0^{y'+y''} g_1^I g_2^{\tilde{B}-v} g_3^s$, so the balance is updated to $\tilde{B} - v$ and the opening is unchanged. The supplier then sends A to the user with y'' and e. When the user receives the A, y'' and e from the supplier, he verifies the correctness of A by (2) with $c = c_0 g_0^{y''} g_2^{-v}$ and updates his internal state to $(A, e, y' + y'', I, \tilde{B} - v, s)$.

Recharging. Recharging operation is the same as charging operation except that the updated balance is increased rather than decreased. This could be achieved by multiplying a commitment with a positive balance, which means that the supplier will compute the commitment as $c = c_0 * g_0^{y''} g_2^{+v}$ to increase the balance by v. In addition, the zero knowledge proof $PK3$ in Recharging phase is also the same as $PK2$ except that the balance checking is rewritten into $D \geq \tilde{B} + v \geq 0$.

Statement. When the user wants to increase the balance in his account, the user could go to the information center and top-up money into his account so that the balance in this account could be reset to the default deposit D. If the state stored in the user's device is $(\tilde{A}, \tilde{e}, \tilde{y}, I, \tilde{B}, \tilde{s})$, he needs to disclose his identity I and pay $D - \tilde{B}$ to the supplier. Besides, he randomly picks two numbers y' and s, computes and sends $c_0 = g_0^{y'} g_3^s$ to the supplier with $\tilde{s}$, I, $\tilde{B}$ and a zero knowledge proof $p = PK4\{(\tilde{A}, \tilde{e}, \tilde{y}, y', s) : c_0 = g_0^{y'} g_3^s \wedge \hat{s}(\tilde{A}, w g^{\tilde{e}}) = \hat{e}(g g_0^{\tilde{y}} g_1^I g_2^{\tilde{B}} g_3^{\tilde{s}}, g)\}$. If $\tilde{s}$ has not been used before, the supplier checks the correctness of the proof p and computes the signature $A = (c_0 g g_0^{y''} g_1^I g_2^D)^{\frac{1}{e+r}}$ where y'' and e are freshly generated random numbers. Then, the supplier will send it with y'' and e to the user. After checking the signature with (2) where $c = c_0 g_0^{y''} g_1^I g_2^D$, the user updates the tuple in his device as $(A, e, y' + y'', I, D, s)$ and completes the statement.

C. Assumptions

Complex cryptographic primitives are used in the protocol. Additionally, the protocol requires redundancy checking, infinite set maintenance and algebra calculation. Modeling these complex operations could easily lead to non-termination of the verification process. Thus we make some assumptions below.

The cryptographic primitives are prefect. For commitment scheme, a commitment could not be opened without knowing the its opening and the opening could not be computed from the commitment. For zero-knowledge proofs, we assume that they will not cause any information leakage for the secret values and they can prove what they intended to prove. For BBS+ signature, signature could not be forged without the signing key.

The supplier will not accept a same opening value twice. This assumption ensures that no user could use the opening value and the signature issued from the supplier twice. The reasons are as follows. The zero-knowledge proofs make sure that the opening value is embedded in the commitment. Besides,

the commitment scheme requires that no other opening value could be forged. When the supplier receives a non-duplicated opening value, if the verification is passed, the opening value and the signature should never be used before.

Algebraic operations on random numbers never introduce duplicated values. In the protocol, add function is used to compute the sum of two numbers, which is used in cryptos such as BBS+ signature and commitment. In symbolic verification, we say two terms are equal when they are structurally equivalent. Since we only apply add function to random numbers in the protocol, if algebraic operations on random numbers never introduce duplicated values, two add results should be equal whenever they are structurally equivalent, which ensures the correctness of the symbolic verification in this paper.

Balance bound checking in zero knowledge proofs are not considered. During the charging and recharging, balance checking is implicitly checked in the zero knowledge proofs $PK2$ and $PK3$. Since checking the explicit values in symbolic verification tools are not possible and bounding the balance or not will not affect the verification results, we omit the balance bound checking in $PK2$ and $PK3$.

D. Primitives Modeling

According to the applications of the primitives and the structure of the protocol, we modeled the primitives as follows:

Commitment. Two forms of commitments are generated in the protocol such as $g_0^y g_3^s$ and $g_0^y g_1^I g_2^B g_3^s$. We modeled them as $resC(y, s)$ and $chtC(y, I, B, s)$ in the tools. When the supplier receives them, he computes the signature A accordingly into a consistent representation.

Zero Knowledge Proof. In this protocol, four zero knowledge proofs are used such as $PK1$, $PK2$, $PK3$ and $PK4$. Since the balance bound checking is omitted in $PK2$ and $PK3$, they become identical so we merged them into one zero knowledge proof of $PK23$. For $PK1$, the prover provide a proof knowledge $regZK(c_0, y', s)$ with a verification function to check the correctness of c_0

$$PK1(resC(y', s), regZK(resC(y', s), y', s)) = true.$$

In $PK23$, the prover's secrets are $(\tilde{A}, \tilde{e}, \tilde{y}, I, \tilde{B}, y', s)$ and the known information to the supplier are $(c_0, \tilde{s}, r)$, so he provide a proof of $chtZK(c_0, \tilde{A}, \tilde{e}, \tilde{y}, I, \tilde{B}, y', s)$ along with a verification function

$$\begin{aligned} PK23(chtC(y', I, \tilde{B}, s), \tilde{s}, r, chtZK(chtC(y', I, \tilde{B}, s), \\ sysA(\tilde{y}, I, \tilde{B}, \tilde{s}, \tilde{e}, r), \tilde{e}, \tilde{y}, I, \tilde{B}, y', s)) = true, \end{aligned}$$

which checks if the c_0 and $\tilde{A}$ are correctly formed with respect to $(\tilde{e}, \tilde{y}, I, \tilde{B}, y', s)$. For statement sub-routine, the proof knowledge $zk = stmZK(c_0, \tilde{A}, \tilde{e}, \tilde{y}, y', s)$. The verifier check the zk with $PK4$ to ensure the c_0 and $\tilde{A}$ are correct.

$$\begin{aligned} PK4(resC(y', s), \tilde{s}, I, \tilde{B}, r, stmZK(resC(y', s), \\ sysA(\tilde{y}, I, \tilde{B}, \tilde{s}, \tilde{e}, r), \tilde{e}, \tilde{y}, y', s)) = true. \end{aligned}$$

BBS+ signature with bilinear pairing. We model the BBS+ signatures in a systematic form as $sysA(y, I, B, s, e, r)$ in which y is the addition of two nonces, I is the identity of the user, B is the current balance in his account, s is the opening of the commitment encoded in the signature, e is the random number chosen by the supplier and r is the private signing key of the supplier. wg^e is modeled by $compose(sysgr(r), e) = sysger(e, r)$ and $w = sysgr(r)$ is a public information. We defined an $extract$ function to model the behavior of (2) as

$$\begin{aligned} extract(sysA(y, I,D, s, e, r), sysger(e, r)) \\ = syse(sysall(y, I, D, s)) \end{aligned}$$

in which the e and r is eliminated because of the bilinear function is used. Thus the user could compute the value of $syse(sysall(y, I, D, s))$ directly.

III. Tools

Nowadays, many symbolic tools, such as Scyther [5], Tamarin [6], ProVerif [7], StatVerif [8] have been developed for automatic analysis of security protocols using different approaches. They have different capabilities for analyzing different protocols with respect to different properties.

Scyther [5] is a tool based on the strand space [13] and the Athena [9] but extends them with trace patterns to reduce the search space. Although unbounded verification could be achieved by Scyther for some protocols, Scyther sometimes bounds the session number to ensure termination of the verification. Additionally, stateful protocol verification and privacy properties are not supported in Scyther.

Tamarin [6], [14] uses multiset rewriting [15] to specify the adversary's capabilities together with a guarded fragment [16] of first-order logic for security properties and equational theories for algebraic properties. Due to the fact that multiset rewriting rules can be directly specified in a model to represent the execution state of the protocol, Tamarin becomes a powerful tool for verifying stateful protocols. Additionally, session indexes can be specified in the query so that authentication properties and secrecy properties with event ordering could be checked in Tamarin.

ProVerif [7] is developed actively since 2001, which uses horn clauses to represent the adversary's capability and backward deduction to check for secrecy. Over approximation on generated session nonces is deployed to limit the searching space but also leads to false attacks. By combining the secrecy checking with inserting special events which indicates the begin and end of the protocol execution [17], authentication checking is then allowed in ProVerif. Blanchet et al. later extended ProVerif with observational equivalence checking [10] and strong secrecy checking [18] so that privacy leakage can be found in protocols.

StatVerif [8] later extends ProVerif with the global mutable state. It could handle protocol with explicit global state by converting the processes into a set of clauses upon which

Property	Scyther	Tamarin	ProVerif	StatVerif
Secrecy	UB/B	UB	UB	UB
Authentication	UB/B	UB	UB	N.A.
Stateful Verification	N.A.	Infinite	Weak	Explicit
Explicit Event Index	N.A.	Supported	N.A.	N.A.
Strong Secrecy	N.A.	N.A.	Supported	N.A.
Observational Equivalence	N.A.	N.A.	Supported	N.A.

TABLE I. TOOL COMPARISON. (UB : UNBOUNDED; B : BOUNDED; N.A. : NOT AVAILABLE.)

ProVerif could verify. In the meanwhile, further abstractions are needed for verifying protocol with infinite state spaces.

Comparison. The differences of these tools are summarized in Table I. As the electric charging protocol [1] analyzed in this paper is a stateful protocol and it uses the commitment scheme which requires explicit event index ordering, Tamarin is the best candidate for analyzing secrecy and authentication properties. On the other hand, strong secrecy and observational equivalence are required for checking privacy properties in this protocol. ProVerif is the only tool that supports these features. Thus, we integrate the verification capacities from Tamarin and ProVerif to give a thorough verification of the protocol.

IV. ANALYSIS IN TAMARIN

A. Abstractions

We abstract the original protocol to ensure the termination of the verification process in Tamarin. Since protocol abstraction can only introduce false alarms, if the protocol is proven as secure after abstraction, its original version should be secure as well.

Fixing the balance for each sub-routines. During the verification in Tamarin, if the balance is allowed to be increased and decreased in the protocol, verification procedure will try to increase its value by infinite times without termination. Because the balance value is not relevant to the secrecy and authentication properties considered in the verification, we fixed the balance along the protocol execution. Since the balance is abstracted to a fixed value, the charging and recharging behavior are then identical. So we merged them into one operation denoted as cht in the model.

Setting the value y'' to 0. In the protocol, y'' is chosen randomly in every sub-routine by the supplier. When the supplier computes the signature, he adds y'' to y' which is encoded in the commitment generated by the user. Since the add function is communicative, we need to reflect the algebra property for add in the model. In the meanwhile, the multiset rewriting rules can only apply to a whole message, so we have to rewrite all the messages where the add function may appear. As a result, the verification process could not terminate because of the complexity introduced by the duplicated rules specified in the model. In order to make the model terminable, we set the y'' generated by the supplier to 0. Thus $add(y', y'')$ is equivalent to y' and modeling the behavior of the algebra function add becomes unnecessary. In fact, y'' is a public value as the supplier will send it out in every session, so fixing its value will not affect the verification results for secrecy and authentication checking.

B. Modeling

Multiset rewriting rules are specified in Tamarin to model the protocol execution. For each sub-routine described in Section II-B, we divide the user's part into two phases. One for sending out the commitment and zero-knowledge proofs, and the other for verifying the signature and updating the internal state of the user. After the registration, every user maintains an internal state of $User(A, e, y, I, B, s)$ in which A is the signature issued from supplier, e and y are nonces, I is the identity of the user, B is the balance in the user's account and s is the opening for the user's commitment encoded in A. Registration and charging/recharging behaviors for both participants are shown in Figure 2. Similar behaviors are defined for statement sub-routine. We use *user_reg_0* and *user_reg_1* to represent the user actions during the registration, and use *server_reg* to specify the server's registration behavior. The two user actions in the registration phase are linked by the user state *UserReg*. Charging/recharging and statement phases have similar structures. As can be seen from the figure, a loop exists in the user charging/recharging phase, which means that a user could do charging/recharging for infinite times. In Tamarin, $pk1$, $pk23$, $pk4$ and signature checking are modeled according to Section II-D. In order to check the zero knowledge proofs and signatures, we defined an axiom on function Eq to test the equivalence of two terms

$$Eq(x, y) \Rightarrow x = y.$$

Thus $Eq(pk1(c, zk), true)$ could be tested along the execution. More details are available in [19].

C. Checking Secrecy and Authentication

Secrecy. One interesting secrecy property required by the electric charging protocol is the conditional secrecy property for the opening value s of the commitment. As the user explicitly sends the opening s out for verification, s will be known to the public. In the meanwhile, s should be kept secret before the user intends to do so.

Definition 1: **(Commitment secrecy).** A nonce s satisfies *commitment secrecy* with respect to the event $open$ if and only if s is secret before event $open$ is engaged.
This property can be specified in Tamarin as follows:

$$\forall I, s, i, j. generate(I, s)@i \wedge know(s)@j \\ \Longrightarrow (\exists r. open(I, s)@r \wedge r < j)$$

in which i, j and r are session ids, I is the identity of the user, and s is the value that should satisfy *commitment secrecy* regarding to $open$ event. The formula means whenever the opening value s generated by I is known to the adversary, there exists an event $open$ explicitly engaged by I before it is known.

Authentication. As this protocol is a charging protocol, we need to make sure that the correspondences between the users and the supplier are established correctly. We adopt the definition of non-injective agreement and agreement from [20] and formalize them for the protocol in Tamarin. In the protocol, because we deem the supplier as the adversary, the users

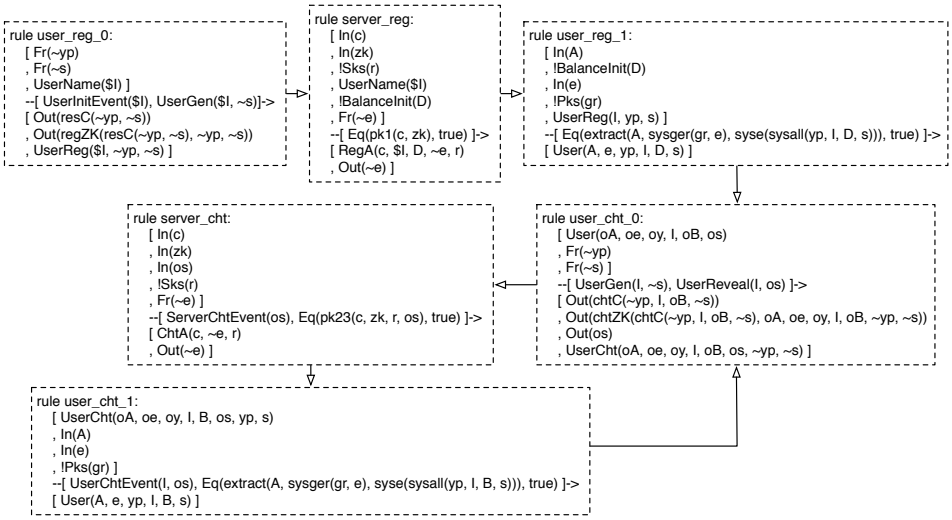

Fig. 2. Modeling in Tamarin

Routine	Property	Result	Time	Step
Charging and Recharging	Secrecy	SAT	1.72s	54
	Non-injective Auth	SAT	7.14s	231
	Injective Auth	SAT	1092.87s	23895
Statement	Secrecy	SAT	1.12s	20
	Non-injective Auth	SAT	1.56s	33
	Injective Auth	SAT	9.09s	251
Charging, Recharging and Statement	Secrecy	N.T.	-	-
	Non-injective Auth	N.T.	-	-
	Injective Auth	N.T.	-	-

TABLE II. VERIFICATION RESULTS FOR TAMARIN : SAT - SATISFIED, N.T. - NON-TERMINATING

need to make sure that the other participant of the protocol is taking the role of supplier. Thus, we formalize the non-injective agreement between the supplier and the user as

$$\forall I, s, i. \mathit{UserCht}(I, s)@i \Longrightarrow (\exists r. \mathit{SupplierCht}(s)@r)$$

which means whenever a user is taking his role for charging or recharging, the supplier is also taking his role in the protocol. Additionally, we formalize the injective agreement as

$$\forall I, s, i. \mathit{UserCht}(I, s)@i \Longrightarrow (\exists r. \mathit{SupplierCht}(s)@r \wedge (\forall j. \mathit{UserCht}(I, s)@j \Rightarrow i = j)).$$

The injective agreement makes sure that for any operation taken by the supplier, there is only one user who is talking to him.

Verification results. We have checked the above properties against different scenarios of the protocol, the verification results are summarized in Table II. All the experiments are conducted under Mac OS X 10.9.1 with 2.3 GHz Intel Core i5 and 16G 1333MHz DDR3. After the registration is finished, any user could do charging/recharging for infinite times with these three properties being preserved. In addition, these properties are also hold for infinite times of statement operations after registration. However, if we check the properties with the charging/recharging and statement sub-routines combined, the verification procedure does not terminate.

V. ANALYSIS IN PROVERIF

A. Abstractions

During the protocol modeling in ProVerif, some abstractions are made to ensure the termination of the verification process.

Fixing the values of y'' and e. In ProVerif, the value of a newly generated nonce depends on two factors: the name of the nonce and the value of the messages received before the generation point. In Section II-B, we have shown that two nonces y'' and e will be generated by the supplier in every session after the supplier receives the commitment, the zero-knowledge proof and an opening value $\tilde{s}$. Additionally, the nonces $\tilde{y''}$ and $\tilde{e}$ generated in the last session are encoded in the zero-knowledge proof, which then leads to the infinite dependency traces of y'' and e. Thus, the verification process cannot terminate. To break the infinite dependency chain, we model y'' and e as two globally shared nonces instead of generating them freshly in every session. Because fixing the value of y'' and e will not affect the users' privacy, this abstraction will not introduce false positives into the verification result.

Setting the value of y'' to 0 for intractability and unlinkability checking. During intractability and unlinkability analysis, the communicative law required by the function add will lead to the non-termination of the verification. Since we could not remove the communicative law which may lead to false negatives, we set $y'' = 0$ so that $add(y', y'')$ is equivalent to y'. Then we could safely eliminate the usage of add with its functionality preserved. As the users' privacy will not be weaken by fixing y'', this abstraction will not introduce any false positives either.

B. Modeling

During the modeling in ProVerif, locations are modeled by public channels and the users' state are passed by the

process arguments to simulate the protocol execution. We have modeled eight processes for four communication sub-routines between the users and the supplier by following the protocol specification described in Section II-B. Since the infinite iterations cannot be specified in ProVerif model, we explicitly call different phases when the current routine is finished. For instance, when we check location privacy, we call $UserCrg$ by passing the user state as arguments to the process. (See more details in [19].)

Routine	Property	Result	Time
Charging	Location Privacy	SAT	131.64s
	Intractability	SAT	13.87s
	Anonymity	SAT	1391.20s
	Strong Anonymity	SAT	1717.25s
	Unlinkability	SAT	5.94s
Recharging	Location Privacy	SAT	132.43s
	Intractability	SAT	14.96s
	Anonymity	SAT	1372.65s
	Strong Anonymity	SAT	1804.23s
	Unlinkability	SAT	6.63s

TABLE III. VERIFICATION RESULTS FOR PROVERIF : SAT - SATISFIED

C. Checking Privacy

Users' privacy is the main purpose of designing this protocol, in which the supplier is the potential adversary for breaking the users' privacy. In this section, we give precise definitions of anonymity and location privacy for the protocol, and we investigate other properties for the protocol as well, including intractability, anonymity, strong anonymity and unlinkability. We present the verification results for all of them. Privacy properties are normally modeled by observational equivalence in the applied pi calculus, which is a widely accepted approach [21], [22], [23]. In the following discussions, we use ECP to represent the well-formed representation [22] for the electric vehicle charging protocol and use Reg, Crg, Rcg and Stm for registration, charging, recharging and statement sub-routines respectively. $Rtn\{n_1/p_1, \ldots, n_m/p_m\}$ means that the routine Rtn parameterized with $p_1, \ldots, p_m$ is instantiated by the values $n_1, \ldots, n_m$. Since the protocol is stateful, $Reg.Crg.Crg$ is different from $Reg.(Crg|Crg)$. Additionally, infinite iterations of the process cannot be specified in ProVerif, so we only give proofs to finite execution iterations with infinite replications and define ECP_{Reg} and ECP_{Crg} as follows

$$\begin{aligned} ECP_{Reg} =& !v(i).Reg\{i/id\}, \\ ECP_{Crg} =& !v(i).Reg\{i/id\}.Crg\{i/id\}. \end{aligned}$$

Similarly, we could define the well-formed protocol for recharging as ECP_{Rcg}.

Location privacy. Location privacy should be guaranteed for the users' behaviors in the stations, which could be specified as

Definition 2: **(Location privacy).** A user A's charging behavior conducted at station X satisfies location privacy if there exists a user B at station Y s.t.

$$\begin{aligned} & C[(Reg\{A/id\}.Crg\{A/id, X/l\} \\ & \quad |Reg\{B/id\}.Crg\{B/id, Y/l\})] \\ \sim\ & C[(Reg\{A/id\}.Crg\{A/id, Y/l\} \\ & \quad |Reg\{B/id\}.Crg\{B/id, X/l\})]. \end{aligned}$$

The stations could be modeled in ProVerif by different channels. Location privacy could also be defined for recharging behaviors similarly. In addition, we also specify *intractability* for users' behaviors as

Definition 3: **(Intractability).** Two subsequent charging behaviors conducted by a user A at station X satisfies intractability if there exists a user B and a station Y s.t.

$$\begin{aligned} & C[(Reg\{A/id\}.Crg\{A/id, X/l\}.Crg\{A/id, X/l\} \\ & \quad |Reg\{B/id\}.Crg\{B/id, Y/l\}.Crg\{B/id, Y/l\})] \\ \sim\ & C[(Reg\{A/id\}.Crg\{A/id, X/l\}.Crg\{A/id, Y/l\} \\ & \quad |Reg\{B/id\}.Crg\{B/id, Y/l\}.Crg\{B/id, X/l\})] \end{aligned}$$

which means the adversary cannot distinguish if a user performs charging at a same station or different stations. Similarly, we can define intractability for recharging behaviors.

Anonymity and strong anonymity. Anonymity should be preserved when the user is doing charging and recharging at stations. Anonymity for charging is defined as follows.

Definition 4: **(Anonymity).** A well formed protocol ECP_{Crg} satisfies anonymity property for a user A's charging behavior if there exists a user B s.t.

$$\begin{aligned} & C[ECP_{Crg}|(Reg\{A/id\}|Reg\{B/id\}).Crg\{A/id\}] \\ \sim\ & C[ECP_{Crg}|(Reg\{A/id\}|Reg\{B/id\}).Crg\{B/id\}] \end{aligned}$$

A stronger notion for anonymity is proposed in [22] which ensures that the adversary cannot tell whether a user A has participated a protocol run or not.

Definition 5: **(Strong anonymity).** A well formed protocol ECP_{Crg} satisfies strong anonymity property for a user A's charging behavior if

$$\begin{aligned} & C[ECP_{Crg}|Reg\{A/id\}] \\ \sim\ & C[ECP_{Crg}|Reg\{A/id\}.Crg\{A/id\}] \end{aligned}$$

Similarly, we could define (strong) anonymity for recharging behaviors.

Unlinkability. In protocol ECP, unlinkability is claimed for charging and recharging operations at stations so that the no adversary, including the supplier, could tell that two operations are initiated by the same user.

Definition 6: **(Unlinkability).** A well formed protocol ECP_{Crg} satisfies unlinkability for a user A's charging behavior if there exists a user B s.t.

$$\begin{aligned} & C[(Reg\{A/id\}.Crg\{A/id\}.Crg\{A/id\})|(Reg\{B/id\})] \\ \sim\ & C[(Reg\{A/id\}.Crg\{A/id\})|(Reg\{B/id\}.Crg\{B/id\})] \end{aligned}$$

Verification results. We have successfully verified location privacy, intractability, anonymity, strong anonymity and unlinkability for users' charging and recharging behaviors in ProVerif. We do the experiments under Mac OS X 10.9.1 with 2.3 GHz Intel Core i5 and 16G 1333MHz DDR3. The verification results are summarized in Table III.

VI. Discussion and Conclusion

In this paper, we presented a verification of an electric vehicle charging protocol proposed by Liu et al. [1] using two most efficient tools. We have checked various security and privacy properties for the protocol, such as secrecy and authentication in Tamarin, and location privacy, intractability, anonymity and unlinkability in ProVerif. Moreover, we have addressed the capabilities of Tamarin and ProVerif and we are the first to combine their verification results for the symbolic analysis of a single protocol.

We have made a few assumptions on the cryptos used in the protocol in the paper. In the future, it is very interesting for us to use CryptoVerif [24], an automatic protocol prover which is sound in proving computational models, to verify the cryptographic primitives used in this protocol.

Acknowledgement. We would like to thank Sjouke Mauw for his support for our research cooperation. Besides, Sun Jun is supported by project "IGDSi1301012" from SUTD.

References

[1] J. K. Liu, M. H. Au, W. Susilo, and J. Zhou, "Enhancing location privacy for electric vehicles (at the right time)," in *Proc. 17th European Symposium on Research in Computer Security (ESORICS)*, ser. LNCS, vol. 7459. Springer, 2012, pp. 397–414.

[2] M. H. Au, W. Susilo, and Y. Mu, "Constant-size dynamic *k*-TAA," in *Proc. 5th Conference on Security and Cryptography for Networks (SCN)*, ser. LNCS, vol. 4116, 2006, pp. 111–125.

[3] R. M. Needham and M. D. Schroeder, "Using encryption for authentication in large networks of computers," *Communications of the ACM*, vol. 21, no. 12, pp. 993–999, 1978.

[4] G. Lowe, "Breaking and fixing the needham-schroeder public-key protocol using fdr," in *Proc. 2nd Workshop on Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, ser. LNCS, vol. 1055. Springer, 1996, pp. 147–166.

[5] C. Cremers, "The Scyther tool: Verification, falsification, and analysis of security protocols," in *Proc. 20th Conference on Computer Aided Verification (CAV)*, ser. LNCS, vol. 5123. Springer, 2008, pp. 414–418.

[6] S. Meier, B. Schmidt, C. Cremers, and D. A. Basin, "The Tamarin prover for the symbolic analysis of security protocols," in *Proc. 25th Conference on Computer Aided Verification (CAV)*, ser. LNCS, vol. 8044. Springer, 2013, pp. 696–701.

[7] B. Blanchet, "An efficient cryptographic protocol verifier based on Prolog rules," in *Proc. 14th IEEE Computer Security Foundations Workshop (CSFW)*. IEEE CS, 2001, pp. 82–96.

[8] M. Arapinis, E. Ritter, and M. D. Ryan, "StatVerif: Verification of stateful processes," in *Proc. 24th IEEE Computer Security Foundations Symposium (CSF)*. IEEE CS, 2011, pp. 33–47.

[9] D. X. Song, S. Berezin, and A. Perrig, "Athena: a novel approach to efficient automatic security protocol analysis," *Journal of Computer Security*, vol. 9, no. 1-2, pp. 47–74, 2001.

[10] B. Blanchet, M. Abadi, and C. Fournet, "Automated verification of selected equivalences for security protocols," *Journal of Logic and Algebraic Programming*, vol. 75, no. 1, pp. 3–51, 2008.

[11] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proc. 11th Annual International Cryptology Conference (CRYPTO)*, ser. LNCS, vol. 576. Springer, 1991, pp. 129–140.

[12] M. Naor, "Bit commitment using pseudorandomness," *Journal of Cryptology*, vol. 4, no. 2, pp. 151–158, 1991.

[13] F. J. T. Fábrega, "Strand spaces: proving security protocols correct," *Journal Computer Security*, vol. 7, no. 2-3, pp. 191–230, 1999.

[14] B. Schmidt, S. Meier, C. Cremers, and D. A. Basin, "Automated analysis of Diffie-Hellman protocols and advanced security properties," in *Proc. 25th IEEE Computer Security Foundations Symposium (CSF)*. IEEE CS, 2012, pp. 78–94.

[15] I. Cervesato, N. A. Durgin, P. Lincoln, J. C. Mitchell, and A. Scedrov, "A meta-notation for protocol analysis," in *Proc. 12th IEEE Computer Security Foundations Workshop (CSFW)*. IEEE CS, 1999, pp. 55–69.

[16] H. Andréka, I. Németi, and J. van Benthem, "Modal languages and bounded fragments of predicate logic," *Journal of Philosophical Logic*, vol. 27, no. 3, pp. 217–274, 1998.

[17] B. Blanchet, "From secrecy to authenticity in security protocols," in *Proc. 10th Symposium on Static Analysis (SAS)*, ser. LNCS, vol. 2477. Springer, 2002, pp. 342–359.

[18] ——, "Automatic proof of strong secrecy for security protocols," in *Proc. 25th IEEE Symposium on Security and Privacy (S&P)*. IEEE CS, 2004, pp. 86–100.

[19] "Models of the electric vehicle charging protocol in ProVerif and Tamarin." [Online]. Available: http://www.comp.nus.edu.sg/~li-li/r/saevcp.html

[20] G. Lowe, "A hierarchy of authentication specification," in *Proc. 10th Computer Security Foundations Workshop (CSFW)*. IEEE CS, 1997, pp. 31–44.

[21] S. Delaune, S. Kremer, and M. Ryan, "Verifying privacy-type properties of electronic voting protocols," *Journal of Computer Security*, vol. 17, no. 4, pp. 435–487, 2009.

[22] M. Arapinis, T. Chothia, E. Ritter, and M. Ryan, "Analysing unlinkability and anonymity using the applied pi calculus," in *Proc. 23rd IEEE Computer Security Foundations Symposium (CSF)*. IEEE CS, 2010, pp. 107–121.

[23] N. Dong, H. L. Jonker, and J. Pang, "Formal analysis of privacy in an eHealth protocol," in *Proc. 17th European Symposium on Research in Computer Security (ESORICS)*, ser. LNCS, vol. 7459. Springer, 2012, pp. 325–342.

[24] B. Blanchet, "A computationally sound mechanized prover for security protocols," in *Proc. 27th IEEE Symposium on Security and Privacy (S&P)*. IEEE CS, 2006, pp. 140–154.